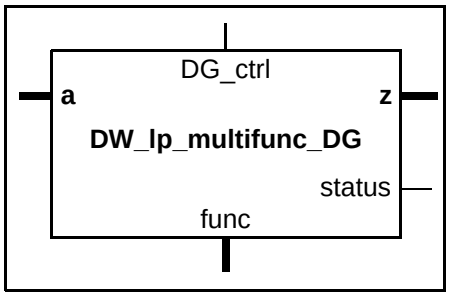
DW_lp_multifunc_DG

Low Power Multi-Function Unit with Datapath Gating

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History



- Has the same functionality as DW_lp_multifunc when the component is in normal operation.
- Consumes less dynamic power than DW_lp_multifunc when disabled (inputs are active, but outputs are not used).
- Parameterized precision of operands and number of functions to be implemented.
- Implements any combination of the following functions:

$$\frac{1}{x}, \frac{1}{\sqrt{x}}, \sqrt{x}, \sin(\pi x), \cos(\pi x), \log_2(x), 2^x$$

- One function is computed at a time.
- All functions produce monotonic results (depends on input range--see Table 1-9 on page 5).
- Shared polynomial approximation unit provides a solution with reduced area.

Note You must use VCS to simulate the DW_lp_multifunc component. Non-VCS simulators are not supported.

Description

DW_lp_multifunc_DG is a fixed point multi-function unit that implements any combination of seven functions: reciprocal, square root, reciprocal square root, sine, cosine, base-2 logarithm and base-2 exponential. Additionally, the component has a control input `DG_ctrl` that disables the component and reduces its dynamic power consumption when the component is not in use, but the inputs are still active. The particular set of functions to be implemented is determined with the *func_select* parameter as a one-hot value. The *op_width* parameter determines the bit-width of input and output values. At any given time the unit computes one function in the set defined by input `func`. An incorrect input value at port `func` is signaled by output `status`.

Table 1-1 Pin Descriptions

Pin Name	Width	Direction	Function
a	(*op_width* + 1) bits	Input	Input data
func	16 bits	Input	Function selection
DG_ctrl	1 bit	Input	Datapath Gating Control ■ 0 = Component is disabled ■ 1 = Normal component operation See
z	(*op_width* + 2) bits	Output	Output data
status	1 bit	Output	Flag of invalid operation

Table 1-2 Parameter Description

Parameter	Values	Description
op_width	3 to 24 Default: 24	Word length of input and output
func_select	1 to 127 Default: 127	Determines the functions to be implemented among the supported functions (one bit for each function).

Table 1-3 Synthesis Implementations

Implementation Name	Implementation	License Feature Required
rtl	Datapath gating close to the main inputs	■ DesignWare (P-2019.03 and later) ■ DesignWare-LP (before P-2019.03)
rtl2[a]	Datapath gating to allow late arrival time of DG_ctrl signal	■ DesignWare (P-2019.03 and later) ■ DesignWare-LP (before P-2019.03)

a. By default, the rtl2 implementation is used for synthesis (see "Datapath Gating Control with DG_ctrl" on page 6).

Table 1-4 Simulation Models

Model	Function
dw/sim_ver/DW_lp_multifunc_DG.v	Verilog simulation model[a] source code You must use VCS to simulate the DW_lp_multifunc_DG component. Non-VCS simulators are not supported.

a. To use this simulation model, the '+v2k' options needs to be used on the VCS command line.

Parameter *func_select* and Input func

The *func_select* is a 7-bit parameter used for the selection of the set of functions to be implemented. The value of I has the range from 1 to 127.

Table 1-5 func_select Parameter Implementation (design) and func Function Selection (use)

func_select Bit	Function Description	func 16-bit select value
2^0	reciprocal, 1/a	0000_0000_0000_0001
2^1	square root of a	0000_0000_0000_0010
2^2	reciprocal square root of a	0000_0000_0000_0100
2^3	sine, sin(πa)	0000_0000_0000_1000
2^4	cosine, cos(πa)	0000_0000_0001_0000
2^5	base-2 logarithm, $\log_2$ a	0000_0000_0010_0000
2^6	base-2 exponential, 2^a	0000_0000_0100_0000
	(2^7 - 2^{15}) reserved for future support	reserved for future support

For example, if only a reciprocal function is required, *func_select* needs to be 1 (16'h0001).

If reciprocal, reciprocal square root and base-2 logarithm functions are needed, *func_select* should be 37 (16'h0025).

If all seven functions are implemented, *func_select* is 127 (16'h007f).

During operation, the input port `func` specifies the single function that is to be computed at that time. The `func` port is a 16-bit input port, and it receives a one-hot encoded value. The valid one-hot codes for `func` are defined in Table 1-5. If `func` receives a code for a function was not implemented by the *func_select* parameter, the `status` flag will be turned on. If `func` does not receive a valid one-hot encoded value, it turns `status` flag on as well. However, if only one function was implemented (*func_select* took value of either 1, 2, 4, 8, 16, 32, or 64), DW_lp_multifunc_DG becomes independent of the input `func`, and `status` flag will not be turned on.

Input and Output Ranges of Functions

Valid input ranges of the seven functions are described in Table 1-6. If the input value is out of range, it turns `status` flag on indicating an incorrect output value.

Table 1-6 Input and Output Ranges

func_select Bit	Function Description	Input Range	Output Range
2^0	reciprocal, 1/a	$1 \le a < 2$	$0.5 < z \le 1$
2^1	square root of a	$0.25 \le a < 1$	$0.5 \le z < 1$
2^2	reciprocal square root of a	$0.25 \le a < 1$	$1 < z \le 2$
2^3	sine, sin(πa)	$0 \le a < 2$	$-1 \le z \le 1$

Table 1-6 Input and Output Ranges (Continued)

func_select Bit	Function Description	Input Range	Output Range
2^4	cosine, cos(πa)	$0 \le a < 2$	$-1 \le z \le 1$
2^5	base-2 logarithm, $\log_2 a$	$1 \le a < 2$	$0 \le z < 1$
2^6	base-2 exponential, 2^a	$0 \le a < 1$	$1 \le z < 2$

Input and Output Format of Functions

Since valid input and output ranges are different among the seven functions, the number formats for the input and output ports vary. Therefore, the input format is one bit wider than *op_width* and the output format is two bits wider. The input format consists of one-bit integer part and *op_width*-bit fractional part. The output format has a two-bit integer part and *op_width*-bit fractional part. For example, Table 1-7 shows the input and output formats with *op_width* = 24. For sine and cosine functions, the output value is represented in two's complement to express signed numbers.

Table 1-7 Input and Output Ranges

func_select	Function Description	Input Format	Output Format
2^0	reciprocal, 1/a	$1.a_{23}a_{22} \ldots a_1 0$	$0\ z_{24}.z_{23}z_{22} \ldots z_1 z_0$
2^1	square root of a	$0.a_{23}a_{22} \ldots a_1 a_0$	$00.1\ z_{22} \ldots z_1 z_0$
2^2	reciprocal square root of a	$0.a_{23}a_{22} \ldots a_1 a_0$	$z_{25}z_{24}.z_{23}z_{22} \ldots z_1 0$
2^3	sine, sin(πa)	$a_{24}.a_{23}a_{22} \ldots a_1 0$	$z_{25}z_{24}.z_{23}z_{22} \ldots z_1 0$
2^4	cosine, cos(πa)	$a_{24}.a_{23}a_{22} \ldots a_1 0$	$z_{25}z_{24}.z_{23}z_{22} \ldots z_1 0$
2^5	base-2 logarithm, $\log_2 a$	$1.a_{23}a_{22} \ldots a_1 0$	$00.z_{23}z_{22} \ldots z_1 z_0$
2^6	base-2 exponential, 2^a	$0.a_{23}a_{22} \ldots a_1 a_0$	$01.z_{23}z_{22} \ldots z_1 0$

Calculation Examples

Table 1-8 shows the examples of the calculation results with *op_width* = 8 and *func_select* = 127. Numbers for input and output in Table 1-8 are binary numbers.

Table 1-8 Calculation Examples (*op_width* = 8, *func_select* = 127)

func	Expression	Input	Output
16'h0001	1/1.0111111 = 0.10101011	1_0111_1110	00_1010_1011
16'h0002	SQRT(0.11011101) = 0.11101101	0_1101_1101	00_1110_1101
16'h0004	1/SQRT(0.11011101) = 1.0100011	0_1001_1101	01_0100_0110
16'h0008	sin(1.1101011π) = -0.011111	1_1101_0110	11_1000_0010
16'h0010	sin(1.1101011π) = 0.110111	1_1101_0110	00_1101_1110
16'h0020	$\log_2$(1.1000101) = 0.10011111	1_1000_1010	00_1001_1111

Table 1-8 Calculation Examples (*op_width* = 8, *func_select* = 127) (Continued)

func	Expression	Input	Output
16'h0040	$2^{0.11100111} = 1.1101111$	1_1100_1110	01_1101_1110

Error Range and Monotonicity

The calculated output value of all functions guarantees 1-ulp error range. Each function has a different ulp position depending on the valid input range. The reciprocal, sine, cosine and base-2 logarithm functions have the input range larger than 1, and the ulp position has weight $2^{-(op\_width-1)}$. The ulp position of square root, inverse square root and base-2 exponential has weight 2^{-op_width}.

The DW_lp_multifunc_DG component produces monotonic results. The input ranges under which monotonic results are produced are shown in Table 1-9.

Table 1-9 Input Range for Monotonic Results

	1/x	sqrt(x)	1/sqrt(x)	sin(x)	cos(x)	log2(x)	exp2(x)
op_width Input Range	All values of *op_width*	All values of *op_width*	All values of *op_width*	*op_width* <= 17	*op_width* <= 17	All values of *op_width*	All values of *op_width*

Verilog Description Example

If all functions except sine and cosine functions are implemented with *func_select* = 103 (7'b1100111) and the output needs to produce the result of the "square root" operation, the Verilog expressions needs to be described as follows.

```
DW_lp_multifunc_DG #(8, 103) U1 (
    .a(9'b011011101),
    .func(16'h0002),
    .DG_ctrl(1'b0)
    .z(z),
    .status(status)
);
```

In this case, `z` becomes 10'b0011101101 as shown in Table 1-8 on page 4 and `status` becomes 0. However, if the above module tries to calculate the sine function with `func` = 16'h0008, it signals the `status` flag, meaning that it is an invalid operation because DW_lp_multifunc_DG does not have the netlist for the sine function with *func_select* = 103 (7'b1100111).

Datapath Gating Control with DG_ctrl

For DW_lp_multifunc_DG and other combinational components that have the datapath gating feature, the `DG_ctrl` port is provided to control datapath gating.

When `DG_ctrl` = 1, the component behaves as expected according to activity on the input ports.

When `DG_ctrl` = 0:

- The component is disabled and internal gates are totally or partially isolated to block propagation of switching activity inside the component. This makes the component less sensitive to switching activity on the main ports and reduces dynamic power consumption.
- Values at the output ports are not defined.
- Simulation models set 'X' values at the output ports.

The implementations for DW_lp_multifunc_DG (see Table 1-3 on page 2) perform datapath gating differently:

- The rtl implementation places datapath gating as close as possible to the input ports to maximize dynamic power savings when `DG_ctrl` = 0. However, if the `DG_ctrl` signal arrives later than the data inputs, timing is degraded and area is increased to recover timing, which can increase power.
- The rtl2 implementation places datapath gating near the middle of the component. This approach is less sensitive to the arrival time of `DG_ctlr` and has a better chance of meeting timing and still providing dynamic power savings.

By default, the synthesis tool uses the rtl2 implementation, but you can override that. If timing constraints are loose or you know that the signal driving the `DG_ctrl` port arrives at the same time as other input signals, greater power savings can be attained by using the rtl implementation. You can make the override on a global level or on a case-by-case basis, as explained next.

To use the rtl implementation globally, you can disable the rtl2 implementation as follows:

- Design Compiler (before version P-2019.03):

```
set_dont_use {dw_minpower.sldb/DW_fp_div_DG/rtl2}
```

- Design Compiler (P-2019.03 and later)

```
set_dont_use {dw_foundation.sldb/DW_fp_div_DG/rtl2}
```

- Fusion Compiler:

```
set_synlib_dont_use {dw_foundation/DW_fp_div_DG/rtl2}
```

To use the rtl implementation for specific instantiated components, use the set_implementation command:

```
set_implementation U1 rtl
```

Related Topics

- Math – Arithmetic Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_lp_multifunc_DG_inst is
      generic (
        inst_op_width : POSITIVE := 24;
        inst_func_select : POSITIVE := 127
        );
      port (
        inst_a : in std_logic_vector(inst_op_width downto 0);
        inst_func : in std_logic_vector(15 downto 0);
        inst_DG_ctrl : in std_logic;
        z_inst : out std_logic_vector(inst_op_width+1 downto 0);
        status_inst : out std_logic
        );
    end DW_lp_multifunc_DG_inst;

architecture inst of DW_lp_multifunc_DG_inst is

begin

    -- Instance of DW_lp_multifunc_DG
    U1 : DW_lp_multifunc_DG
    generic map (
          op_width => inst_op_width,
          func_select => inst_func_select
          )
    port map (
          a => inst_a,
          func => inst_func,
          DG_ctrl => inst_DG_ctrl,
          z => z_inst,
          status => status_inst
          );

end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_lp_multifunc_DG_inst( inst_a, inst_func, inst_DG_ctrl, z_inst, status_inst );

parameter inst_op_width = 24;
parameter inst_func_select = 127;

input [inst_op_width : 0] inst_a;
input [15 : 0] inst_func;
input inst_DG_ctrl;
output [inst_op_width+1 : 0] z_inst;
output status_inst;

    // Instance of DW_lp_multifunc_DG
    DW_lp_multifunc_DG #(inst_op_width, inst_func_select) U1 (
                .a(inst_a),
                .func(inst_func),
                .DG_ctrl(inst_DG_ctrl),
                .z(z_inst),
                .status(status_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
January 2020	DWBB_201912.1	■ Corrected port names to be lowercase in Table 1-1 on page 2, examples, and format presentations
March 2019	DWBB_201903.0	■ Added "Datapath Gating Control with DG_ctrl" on page 6 ■ Clarified some information about minPower
July 2018	DWBB_201806.1	■ For STAR 9001366625, added a note about simulator support on page 1 and in Table 1-4 on page 2 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com